United States Patent Application

of

Ann M. Wollrath, Peter C. Jones, James H. Waldo, Robert W. Schiefler

for

REMOTE OBJECT ACTIVATION IN A DISTRIBUTED SYSTEM

LAW OFFICES
FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER, L. L. P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

BACKGROUND OF THE INVENTION

The present invention relates generally to distributed computer systems, and more specifically, to managing and activating objects in distributed computer systems.

A distributed computer system is a network of computer processors that although geographically separated, are linked together functionally. It is often desirable to execute a computer program, or portions of a computer program, simultaneously across several computer processors in the distributed system. In such an environment, protocols coordinating the various portions of the program(s) are necessary.

Distributed computing systems executing object-oriented programming models are known. Essentially, in these systems, programs are written as an aggregation of objects, each of which may reside on and be executed on a different computer in the distributed system.

Typically, in an object-oriented distributed system, a local computer system, called the client, may access objects on remote computer systems. If the objects to be accessed on the remote computer system take up processor resources, i.e., if they consume physical or virtual memory and have a thread of control, they are said to be "active." Examples of such active objects include running programs or objects that are part of active programs. Such objects are always taking up resources from the physical machine, even when they are not doing active work on behalf of themselves or at the request of some other object.

A "passive" object, on the other hand, refers to a presently non-active object on the remote computer. If a passive object is "activatable," it may, at the request of the client computer system, be brought into an active state. Objects may be passive simply because they have never

LAW OFFICES
FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER, L.L.P.
1300 I STREET, N.W.
WASHINGTON, D.C. 20005
202-408-4000

been instantiated. Alternatively, to save system resources, active objects may be de-activated and become passive. In particular, for active objects that have become quiescent, it may be advantageous for the computer to save the state information of the object to a stable storage medium, such as a magnetic disk, and release any memory or threads of control associated with the object. The de-activated object does not take up physical or virtual memory and is not associated with a control thread, although it continues to exist and may be made active when called.

One known distributed system capable of activating objects is the object management groups Common Object Request Broker Architecture (CORBA) system. In the CORBA system, remote objects are always considered by the client to be potentially passive, and thus activatable, regardless of whether the object is actually active or passive. Additionally, although some objects at a remote system may be similar to one another, and capable of benefiting from a sharing of common resources, CORBA does not provide for the associating of similar objects.

There is, therefore, a need for a distributed system object management architecture that solves the above mentioned limitations found in the prior art.

SUMMARY OF THE INVENTION

Objects and advantages of the invention will be set forth in part in the description which follows, and in part will be obvious from the description, or may be learned by practice of the invention. The objects and advantages of the invention will be realized and attained by means of the elements and combinations particularly pointed out in the appended claims.

LAW OFFICES
FINNEGAN, HENDERSON,
FARABOW, GARRETT
& DUNNER, L.L.P.
1300 I STREET, N.W.
WASHINGTON, D.C. 20005
202-408-4000

To achieve the objects and in accordance with the purpose of the invention, as embodied and broadly described herein, a first aspect of the present invention includes a method of calling a remote object by a process comprising the steps of: (1) calling the remote object directly using a first address in a faulting remote reference to the remote object when the reference refers to an active instance of the remote object; and (2) calling an activator object using a second address in the faulting remote reference to perform activation of the remote object when the reference to the remote object does not refer to an active instance of the remote object. In an alternative aspect, a computer readable medium contains instructions for performing similar steps.

A second method consistent with the present invention includes a method of handling an object call at a remote site for a remote object, the method comprises the steps of: (1) determining whether a first predefined group of objects corresponding to the called remote object is active; (2) activating the remote object within the first group when the determining step determines that the first group is active; and (3) creating a second group of objects and activating the remote object within the second group when the determining step determines that the first group is not active. As with the first method, an alternative aspect includes a computer readable medium containing instructions for performing similar steps.

Still further, a distributed computer system consistent with the present invention includes a plurality of elements, including first and second computers. The second computer, in particular, receives requests for remote objects from the first computer and executes an object activator performing the steps of: (1) determining whether a first predefined group of objects corresponding to the requested remote object is active; (2) activating the requested remote object

LAW OFFICES
FINNEGAN, HENDERSON,
FARABOW, GARRETT
& DUNNER, L.L.P.
1300 I STREET, N.W.
WASHINGTON, D.C. 20005
202-408-4000

within the first group of objects when the determining step determines that the first group of objects is active; and (3) creating a second group of objects and activating the requested remote object within the second group of objects when the determining step determines that the first group of objects is not active.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate several embodiments consistent with this invention and, together with the description, help explain the principles of the invention. In the drawings,

Fig. 1 is a high-level block diagram illustrating interaction of hardware and software components in an exemplary distributed computer system consistent with the present invention;

Fig. 2 is a block diagram illustrating software entities located on a local computer; and

Fig. 3 is a block diagram illustrating software entities located on a remote host computer; and

Fig. 4 is a flow chart illustrating steps consistent with the present invention.

DESCRIPTION OF THE PREFERRED EMBODIMENT

A distributed computer system and related methods consistent with the present invention are described herein. The distributed computer system uses a single interface at the client site to handle calls to call both active and activatable remote objects. Further, remote objects are

LAW OFFICES
FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER, L.L.P.
1300 I STREET, N.W.
WASHINGTON, D.C. 20005
202-408-4000

aggregated into common groups of objects, thereby providing greater security between objects of disparate groups and efficiency between related objects of the same group.

Wherever possible, the same reference numbers will be used throughout the drawings to refer to the same or like parts.

Fig. 1 is a high-level block diagram illustrating interaction of hardware and software components in an exemplary distributed computer system. Computer 102 includes a computer hardware/processing section 107 executing programs from memory section 103. Memory 103 is preferably a random access memory, but may additionally or alternatively include other storage media such as magnetic or optical disks.

Memory 103 stores one or more computer procedures 104a, 104b, and 104c, such as, for example, a computer program, thread, or object. Computer threads 104a and 104b are programs comprised of Java bytecodes executing through a Java virtual machine 105. The virtual machine is itself a process that when executed on computer 102, translates threads 104a and 104b into computer instructions native to computer hardware 107. In this manner, virtual machine 105 acts as an interpreter for computer hardware 107. In contrast to threads 104a and 104b, program 104c uses instructions native to computer hardware 107, and thus does not require virtual machine 105.

Computer 102 is connected via network 120 to computer 112. Computer 112 includes components similar to those of computer 102, and will therefore not be described further. Although the simple network described above includes only two computers, networks of many

LAW OFFICES
FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER, L.L.P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

computers, or even networks of many thousands of computers, such as the Internet, may be used to implement the concepts of the present invention.

Throughout the remainder of this disclosure, computer system 102 will be described as the requester of remote objects. Computer system 112 executes the remote objects and returns results to computer 102. Although not explicitly shown, a plurality of computer systems 112 may execute multiple objects for a single host computer 102.

Fig. 2 is a block diagram illustrating software entities located on computer 102.

Process 202 is a program active on computer 102, such as process 104 in Fig. 1. As shown, process 202 includes a plurality of bytecodes that may be translated from instructions written in the Java programming language, including instruction 203, which is an invocation to a method residing in an object on remote computer 112. The method invocation is preferably defined to be handled by local proxy object 205, which functions as an interface for remote object calls from computer 102 and hides the remote calling protocol from the invocating process.

Proxy 205 may assume one of multiple implementations depending on the status of the object being referenced; such as whether the object is active or activatable (i.e., presently passive). When called by process 202, proxy 205 packages the call using the appropriate implementation and forwards it to remote computer 112. Results received from the remote computer, such as results from the method invocation, are passed back through proxy 205 to process 202.

As described in more detail below, proxy 205 enables process 202 to make a single method invocation for both active and activatable objects. In other words, process 202 is not required to monitor whether a remote object is active or activatable.

Activation of remote objects by proxy 205 is implemented through an object reference known as a faulting remote reference, illustrated by reference 210. For each remote object, faulting remote reference 210 is used to "fault in" the object's reference upon the first method invocation to the object. Faulting remote reference 210 includes a persistent handle (an activation identifier) 211 and a transient remote reference 212. Both persistent handle 211 and transient remote reference 212 are obtained from the remote computer corresponding to the remote object, and contain address information for contacting the remote computer, such as the appropriate network address and port number, and address information more specific to the remote object being referred. Persistent handle 211 is the more general reference and references an activator entity (described in more detail below) at the remote host. Reference 212 is the actual "live" reference to the active remote object, and is used to contact the remote object directly.

In operation, upon invocation of a method requiring a remote object, proxy 205 checks reference 210. A null value in "live" reference 212 indicates that the remote object may become passive (i.e., it is not an active-only object), and proxy 205 uses activation identifier 211 to contact an activator entity at the remote site. If reference 212 is not null it will point directly to the remote object. This indicates an active remote object, which proxy 205 contacts directly.

LAW OFFICES
FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER, L.L.P.
1300 I STREET, N.W.
WASHINGTON, D.C. 20005
202-408-4000

Fig. 3 is a block diagram illustrating software entities located on host computer 112. As mentioned previously, host computer 112 is contacted by the client using either the activation identifier reference 211 or "live" reference 212. Activation identifier reference 211 references object activator 302, which supervises object activation on the host. Activator 302 functions as: (1) a database that maps activation identifiers 211 to the information necessary to activate an object (e.g., the object's class, the location-- a URL-- from where the class can be loaded, specific data the object may need to bootstrap, etc.); (2) a database for tracking the current mapping of activation identifiers to active objects; and (3) a manager of Java virtual machines.

Activatable objects are defined by the designer to exist as a member of a group of objects, such as group 305. The designer preferably assigns objects to particular groups so that objects within a group are designed to interact with one another. For example, objects within a group should have a relationship of mutual trust strong enough to allow them all to run within a single Java virtual machine. Once assigned to a group, objects stay within that group.

Activation entity 304 manages object group 305. In particular, activation entity 304 activates passive objects and creates objects pursuant to requests from object activator 302, and returns a reference to the corresponding activated object to object activator 302. To activate a quiescent object within group 305, activation entity 304 allocates the appropriate operating system resources (memory, process, or thread allocation) and starts up the object. After activating an object, activation entity 304 passes information to object activator 302 describing the way in which the object is to be reached for further communication. Object activator 302 may then forward this information to proxy 205, which appropriately updates faulting remote

LAW OFFICES
FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER, L.L.P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

reference 210. If an object later de-activates, or is de-activated, object activator 302 similarly communicates with proxy 205 to update the faulting remote reference.

Preferably, one activation entity 304 exists per each active Java virtual machine.

Fig. 4 is a flow chart further illustrating steps consistent with the present invention. Upon invocation of a remote object by computer 102, proxy 205 determines if the transient remote reference (the "live" reference) 212 at computer 102 is present, i.e., if it is not null, and proxy 205 contacts the active remote object (steps 402, 404). Otherwise, proxy 205 uses persistent handle 211 within reference 210 to contact object activator 302 (steps 402, 406). Object activator 302 uses information within reference 210 to determine if an object group corresponds to the invocated object (step 408). If an appropriate group is already active, an activation request for the called object is forwarded to the appropriate activation entity (steps 408, 410), and the object is activated (step 411). Otherwise, the activator object first creates a new virtual machine and a new activation entity (steps 408, 412), and then forwards the activation request to the newly created activation entity, (step 414), at which point the object is activated (step 415). In response to forwarding an activation request to an activation entity, the object activator will usually receive an updated network address and port number, which it forwards to proxy object 205 (step 416).

As described above, object groups such as group 305 form the basic unit of object activation. Object activator 302 and activation entities 304 manage the object groups, such that if a group has not been activated then a call to any object of an object group will cause the activation of that object group and the called object in a new Java virtual machine.

LAW OFFICES
FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER, L.L.P.
1300 I STREET, N. W.
WASHINGTON, D. C. 20005
202-408-4000

Clustering objects within an object group on a single Java virtual machine allows related objects to share an address space, which in turn allows close communication between the objects. Objects in different groups, on the other hand, are in different Java virtual machines and thus have a much stronger security separation, ensuring that those objects will not interfere, either intentionally or unintentionally, with each other.

Further, a single interface is seen by clients attempting to call remote objects. The interface has multiple implementations depending on the status of the object being referenced, allowing for transparent mixing of active and passive (i.e., activatable) objects in the same system, supporting both without requiring that the clients of those objects have any knowledge of whether or not the object is activatable. This interface provides the client the ability to make any calls that are supported by the remote object through a faulting remote reference.

It will be apparent to those skilled in the art that various modifications and variations can be made to the concepts of the present invention and in its construction without departing from the scope or spirit of the invention. Other embodiments of the invention will be apparent to those skilled in the art from consideration of the specification and practice of the invention disclosed herein. It is intended that the specification and examples be considered as exemplary only, with a true scope and spirit of the invention being indicated by the following claims.

LAW OFFICES
FINNEGAN, HENDERSON,
FARABOW, GARRETT
& DUNNER, L.L.P.
1300 I STREET, N.W.
WASHINGTON, D.C. 20005
202-408-4000